

SYSTEM DESIGN TOOLS

The present invention relates to system design tools,
and, in particular, to tools for designing bus-based
5 systems.

BACKGROUND OF THE INVENTION

Integrated circuits are often designed using pre-
designed Intellectual Property (IP) modules. Such
10 modules define all of part of a function or function
within an integrated circuit. The modules are often
supplied as a parameterisable core or as an executable
module generator (for example, a Perl, TCL, or Java
executable that generates a configured HDL (Hardware
15 Description language) instance of the IP module).
Ports on a module either provide a complete set of
connections as specified by the bus/communication
protocol, or are configured by the designer to
implement only the subset of features needed for the
20 application.

In one example of a system design tool, the Altera
NIOS™ System Builder uses a propriotor file format
(labelled ".ptf") file to describe connections on an IP
25 module. Each port of the module can be specified as
bus slave or as bus master. Other parameters such as
module base address, port width, and port name on the
component can be specified. When incorporated into a
design, the module description is used to construct a
30 "bus module" which has appropriate connections to the
module itself.

Configuring each module in a system by hand is a time
consuming and error prone activity. As each module is
35 unaware of the requirements and capabilities of the
other modules to which it is connected it is impossible

to remove unused services or make other optimisations in an efficient way. Indeed, with anything other than the most trivial of systems, the task is practically impossible.

5

In addition, new bus protocols cannot be specified without modifying the design tool significantly.

Accordingly, it is desirable to provide a design tool
10 and method which can overcome these problems.

SUMMARY OF THE PRESENT INVENTION

According to one aspect of the present invention, there
15 is provided a system design method for designing systems having a plurality of components interconnected via a bus, the method comprising defining respective functional representations of system components in the system, defining a functional representation of the bus
20 in the system, each such functional representation including a number of variable parameters, and defining an allowed set of parameters for each system component, in dependence upon the functional representations of the set of system components connected to the bus.

25

BRIEF DESCRIPTION OF THE DRAWINGS

Figure 1 illustrates a bus-based system including a plurality of system components; and
30 Figure 2 illustrates a method embodying one aspect of the present invention;
Figure 3 illustrates an apparatus operable in accordance with a method embodying the present invention;
35 Figure 4 illustrates a functional model of a component;

Figure 5 illustrates part of a method embodying one aspect of the present invention; and
Figure 6 illustrates part of a method embodying one aspect of the present invention.

5

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

Figure 1 illustrates a bus-based system as an example of a system to be designed by a designer using a method and apparatus embodying the present invention. The
10 system includes a bus 2 having a number of data lines B, which interconnects a plurality of system components. The number of data lines can be referred to as the "width" of the bus. The bus could be
15 replaced by a simple point-to-point connection. In the example shown in Figure 1, a CPU 4 and a plurality of peripheral devices C1, C2 and C3 are connected to the bus 2. The components communicate with one another via the bus 2. In a system design tool embodying the
20 present invention, each of the system components CPU 4, C1, C2 and C3, and the bus 2, are presented to the user as functional representations. The functional representation of each component includes a number of parameters which define the operating characteristics
25 of the components. For example, the functional representation of the bus 2 could specify the width of the bus 2, and the protocol used for communicating thereon. Similarly, each of the system components has a number of parameters associated therewith, which
30 parameters define the functional operation of the component concerned.

Types of parameter could usefully include type of bus protocol (for example, AHB, OPB, APB, USB), and type of
35 bus transactions supported (for example, burst, locked, word, half word, byte). General parameters, such as

the ordering of bytes of data (endianness), can also be specified.

As described above, using existing development tools,
5 the system designer must define the parameters in each of the components of the system manually, in dependence upon the combination of the components.

In a system design method embodying the present
10 invention, at least one allowed set of parameter values is automatically computed for the components in the system, in dependence upon the overall system configuration. The allowed set of values is such that the system components, when set up according to the
15 values, are compatible with one another. For example, the bus could be configured with 8, 16, 32 or 64 data lines, and could operate using a certain bus protocol. The CPU 4 could support output data widths of 16 or 32 bits, and the components C1, C2 and C3 could each
20 support data widths of 8 or 16 bits. Accordingly, in accordance with the present invention, an allowed parameter value for the bus width, and hence the data connections to each of the system components, would be 16 bits, since this is the only value common to all of
25 the components.

It will be appreciated that the choice of bus width is a particularly simple example, but the principles are applicable to any particular parameter or group of
30 parameters.

Producing a set of allowed parameters in this way enables the system designer to choose more rapidly the parameter values concerned. For example, if the
35 allowed bus width included multiple values, then the system designer would have to choose the particular

value for the design. However, since the possible values presented to the system designer are reduced from the total number of possibilities, to only those values which are applicable to the system as a whole, the design process is much more rapid than previous methods.

A method embodying the present invention is illustrated in Figure 2, and Figure 3 illustrates an apparatus for carrying out the method. The apparatus shown in Figure 3 includes a processor 20 which is operable to carry out steps in the method in response to user inputs received via at least one input device 22. The processor 20 operates to access data stored in a storage medium 26, such as a hard disk drive (HDD), and/or memory. The processor 20 is connected to a display device 24 to display to the user the results of the operations performed. The processor 20, input device(s) 22, display 24 and storage medium 26 are usefully provided by a personal computer (PC) or workstation. Such a device can be operated on a stand-alone basis or in connection with a network.

The processor 20 also has access to a database 30 which holds library information concerning component representations that have been defined previously. The database could be stored locally on the PC or workstation, or could be provided on a central storage device via a network connection.

The method will be described with reference to Figures 2 and 3. The functional representations of the system components are defined and stored (A) in the database 30, and the components for inclusion in the design are selected (B). The components chosen do not necessarily have to include a bus or other data transfer component.

However, most designs use at least one data transfer component. The design system processor 20 automatically selects at least one allowable set of parameter values for the combination of selected system components.

Figure 4 illustrates a component model for use in a method embodying the present invention. The model includes a component name, and a number of parameter values that define the functional representation of the component concerned.

There are three aspects to a preferred method embodying the present invention. Firstly, there is the definition of the bus/communication protocol. Secondly, there is the association of a port on an IP block with one or more such protocols, and thirdly there is the definition of sets of allowed parameter values for the group of system components concerned. The various stages of such a method will now be outlined.

DEFINING A BUS/COMMUNICATION PROTOCOL

Each definition begins with a prologue defining its name (ie. the "PROTOCOL NAME" in Figure 4).

The protocol definition is made up of 4 sections:

1. parameters
2. operations
3. connections
4. roles

1. Parameters

A number of parameters are associated with a protocol definition. For example:

```
<PARAMETER name 1="wait_states" type="integer"
default="0" range="0.3"/>
```

```
<PARAMETER name 2="data_width" type="integer"
default="8" range="8/16/32"/>
```

5

respectively define the number of wait states and the data width of the bus. These parameters have a type and set of allowed values associated with them.

10 2. Operations

Each component port is defined by the "operations" that pass through it. For example, a port on a memory device would conceptually accept read and write operations. Definition checking can be performed to ensure that ports are connected in a manner that is consistent with each of the sets of allowed values of the parameters. For example, a port that generates 16-bit write operations can only be connected to a port that accepts them. Available operations are described as follows:

20

```
<OPERATION name="master-to-bus"
parameter="data_width"/>
```

```
<OPERATION name="bus-to-slave"
parameter=data_width, wait_states"/>
```

25

```
<OPERATION name="master-bus-request"/>
```

Operations are used to describe logical restrictions on the connection of components. Parameters may be associated with operations to ensure that only ports supporting compatible modes of operation are connected.

30

3. Connections

The goal of a protocol definition is to connect two components together correctly. Connections are therefore defined in terms of, for example, width and the input/output functions that the connection performs. The width of the connection can be set to a

35

default value of "1", or can be specified by a specific parameter value. For example, connections can be described as below:

```

5      <WIRE name="_wdata" width="data_width">
      <OUTPUT-FOR operationName="master-to-bus"/>
      <OUTPUT-FOR operationName="bus-to-slave"/>
      </WIRE>
      <WIRE name="_bus_grant" width="1">
      <INPUT-FOR operationName="master-bus-request"/>
10     </WIRE>

```

The direction (input/output) of a connection is defined with respect to an operation. If a component accepts write operations then data is an input, and conversely
15 if a component generates write operations then data is an output. A wire may be specified as both an input-for and an output to.

4. Roles

20 A component typically implements a standard set of connections as part of a wiring definition and so a number of standard modes of operation, or "roles" are defined within the protocol. These roles specify what operations a port may generate or accept and hence the
25 connected to which it is connected.

```

      <ROLE name="32 bus master">
      <PARAMETER name="addr_width" value="32"/>
      <GENERATE operationName="master-to-bus"/>
      <GENERATE operationName="master-bus-request"/>
30     </ROLE>

```

The definition is completed by the XML closing brace:
 </PROTOCOL>

DESCRIBING A PORT ON AN IP BLOCK

35

A port on an IP block can be defined in two ways. The simplest is to specify a role specified in the protocol definition. The following specifies that a port operates as a 32-bit AHB bus master:

```

5      <PORT name="ahb bus port">
        <PROTOCOL spec="com/altera/AlteraAHB.xml">
          <ROLE roleName="32 bus master"/>
        </PROTOCOL>
      </PORT>

```

10

More complex ports can be described by directly specifying the operations that the port may generate and those that may be accepted. The description below is of a port that "accept" a variety of read operations and can generate a "split master response" operation - the specification details that the port cannot "accept" write transactions:

```

15      <PORT name="complex ahb bus port">
        <PROTOCOL spec="com/altera/AlteraAHB.xml">
          <ACCEPT operation="read"/>
          <ACCEPT operation="burst read"/>
          <ACCEPT operation="locked read"/>
          <GENERATE operation="split master response"/>
          <PARAMETER name="addr_width" value="16/32"/>
20      </PROTOCOL>
        </PORT>
25

```

DETERMINING ALLOWABILITY OF CONNECTIONS

30 In order for a designer to connect two ports together, the following process is used to determine the parameters used for such connection. The process is shown in Figure 5.

35 At step AA, a union of the ACCEPT values from the first port and the GENERATE values of the second port is

constructed. This union represents values that the first port can accept from the set that the second port can generate.

- 5 At step AB, a union of the GENERATE set from the first port and the ACCEPT set of the second port is constructed. Similar to above, the union represents values that the second port can accept from those values which the first port can generate.

10

A common value set comprising the contents of the two unions is constructed (step AC), and examined to determine if it is empty (step AD). If it is empty then the connection cannot be made (step AE). If the

- 15 common value set is not empty, then a working choice of values is made.

The common value set for the ports is examined and optimal values are chosen (step AF).

20

DETERMINING REQUIRED PHYSICAL CONNECTIONS

As illustrated in Figure 6, when the physical connection between the ports on the IP blocks is constructed the list of connections in the protocol definition is examined (steps BA) and any connections required as an "OUTPUT-FOR" or an "INPUT-FOR" the operations carried out between the ports are connected (steps BB and BC).

30

This ensures no unnecessary connections are made between the blocks.

- 35 This mechanism of describing component ports and their connectivity allows components to optimise their implementation based on use (by inspecting the final

"resolved" sets of operation used); this mechanism also allows tools to be constructed to connect IP blocks which are independent of any bus protocol.

- 5 The particular method mentioned above includes specific examples of naming and connection protocols. It will be readily appreciated that the techniques embodying the present invention can be implemented in other specific ways.

10

Methods embodying the present invention are particularly useful when implemented using Altera™ Quartus™ or SOPC Builder™ products.